Chapter V

Broadcast Data Placement over Multiple Wireless Channels

Dimitrios Katsaros
Aristotle University of Thessaloniki, Greece

Yannis Manolopoulos
Aristotle University of Thessaloniki, Greece

Abstract

The advances in computer and communication technologies made possible an ubiquitous computing environment were clients equipped with portable devices can send and receive data anytime and from anyplace. Due to the asymmetry in communication and the scarceness of wireless resources, data broadcast is widely employed as an effective means in delivering data to the mobile clients. For reasons like heterogeneous communication capabilities and variable quality of service offerings, we may need to divide a single wireless channel into multiple physical or logical channels. Thus, we need efficient algorithms for placing the broadcast data into these multiple channels so as to reduce the client access time. The present chapter discusses algorithms for placing broadcast data to multiple wireless channels, which cannot be

coalesced into a lesser number of high-bandwidth channels, assuming that there are no dependencies among the transmitted data. We give an algorithm for obtaining the optimal placement to the channels and explain its limitation since it is computationally very demanding and thus unfeasible. Then, we present heuristic schemes for obtaining suboptimal solutions to the problem of reporting on their implementation cost and their relative performance.

Introduction

The technological achievements in the field of computer communications and the ever-decreasing sizes of wireless devices enabled the proliferation of wireless data applications. Mobile clients equipped with laptops, palmtops, personal digital assistants (PDAs), and other portable devices are able to access a variety of information stored in the databases of servers located in fixed wireline networks. The mobile clients roam inside the coverage area of the wireless network requesting various data; types of data that may be of interest to the clients include stock quotes, weather information, traffic conditions, and airline schedules, to name a few.

There are two basic delivery methods in wireless applications: the unicast (or point-to-point) and broadcast methods. In the former, each client establishes a connection with the server and poses a request; in response, the server sends the requested data to the client using the established connection. This delivery method implements the classic client-server paradigm of communication encountered in traditional wireline networks. The broadcast delivery method (Wong, 1988) differs because all clients monitor the same channel (*broadcast channel* or *downlink channel*) in order to acquire the information transmitted by the server. The contents of the transmission are determined by the server, based either on the estimation of client preferences (pure broadcast systems; Acharya, Alonso, Franklin, & Zdonik, 1995) or on the client requests acquired through *uplink channels* (on-demand broadcast systems; Aksoy & Franklin, 1999).

Although, many current systems are based on unicast delivery, the broadcast method is increasingly appealing. Unicast delivery is a waste of resources because each datum must be transmitted for each client that requests it. Thus, the network and server load increases with every client. In contrast, broadcasting is advantageous because of its excellent scalability, that is, a single broadcast satisfies all pending requests for it. Moreover, wireless environments are characterized by the asymmetry in communication, that is, the broadcast channel capacity is much greater than the uplink channel capacity. Therefore, broadcasting is able to exploit the high bandwidth of the downlink channel. Concrete examples of broadcast delivery include the cache-satellite distribution systems (Armon & Levy, 2004), where satellites broadcast popular data, for example, Web pages, to clients (e.g., humans or proxy servers), and the mobile Infostations (Iacono & Rose, 2000).

Multiple-Channel Broadcast Environments

The most common assumption about broadcasting is that there exists a single physical channel for the broadcasted data. There are, though, many scenarios where a server has access to multiple low-bandwidth physical channels which cannot be combined to form a single high-bandwidth channel (Prabhakara, Hua, & Oh, 2000; Yee & Navathe, 2003). Possible reasons for the existence of multiple broadcast channels include application scalability, fault tolerance, reconfiguration of adjoining cells, heterogeneous client-communication capabilities, and so forth. In the following paragraphs, we give example scenarios of the aforementioned reasons.

- *Application scalability*. Consider an application running on the server that needs to be scaled in order to support a larger number of clients. In this case, it may need to acquire additional physical channels. If these channels are in noncontiguous frequencies, they may have to be treated as separate channels when broadcasting data.
- *Fault tolerance*. Suppose that a transmitting station can have more than one server with a transmission capability, and let A, B and C be servers which broadcast data in three noncontiguous frequency ranges all in the same cell. If servers B and C crash, then frequencies assigned to them should be allocated to server A.
- *Reconfiguration of adjoining cells*. Suppose that there are two adjacent cells whose servers transmit in different frequency ranges and, at some point in time, we decide to "merge" the two cells and use one of the two servers to serve the newly generated cell. Then, the frequency range of the other server should be migrated and added to the residual server. In this case also, the latter server gets multiple physical channels.
- *Heterogeneous clients*. The mobile clients may have heterogeneous communication capabilities, precluding the existence of a single high-speed transmission channel.

There are several concerns related to the exploitation of a multichannel broadcast system. The first consideration is related to the capability of the server to concurrently transmit in all channels; the second is related to the capability of the client to simultaneously listen to multiple channels and also perform instantaneous "hopping" among channels. Since the interest of this chapter is to focus on the data placement problem, we will make simplifying assumptions, considering that the server is able to concurrently transmit to all channels, and the clients are able to listen simultaneously to all channels and perform instantaneous hopping from channel to channel. Moreover, we do not assume any kind of dependencies between broadcasted data. Techniques for broadcast scheduling when there exist dependencies among the data can be found in other chapters of this book.

The two major issues in broadcast dissemination are how and what the server transmits and how the client retrieves. Of particular interest are the solutions that enable the mobile clients to get the disseminated data efficiently, that is, with short access latency and with

minimum power expenditure. The former is quantified by the *query access time*, which is equal to the time elapsed between when the client starts seeking for an item until it gets it. The latter is quantified by the *tuning time*, which is the time the client spends actively listening to the broadcast channel. The access time is directly related to the size of the broadcast. On the other hand, providing information to the clients for *selective auto tuning*, that is, indexes (Imielinski, Viswanathan, & Badrinath, 1997), reduces the tuning time. However, including such information increases the overall size of the broadcast, which in turn increases the access time. The trade-off between these two performance measures is obvious.

The chapter's focus is the data placement issues, so we assume that the clients have complete knowledge of the broadcast schedule. In other words, they know a priori the arrival time and channel of all broadcasted data items. Hence, we are only interested in minimizing the client access time and we do not assume the existence of index packets in the broadcast. Though, in order to make the broadcast schedule "predictable," we are interested in placement schemes, which guarantee that

- the broadcast is cyclic, that is, the schedule has a beginning and an end, and
- the interarrival time between successive transmissions of an item is constant for all the broadcast cycles.

Under the aforementioned assumptions, the problem addressed by this chapter can be captured by the following question:

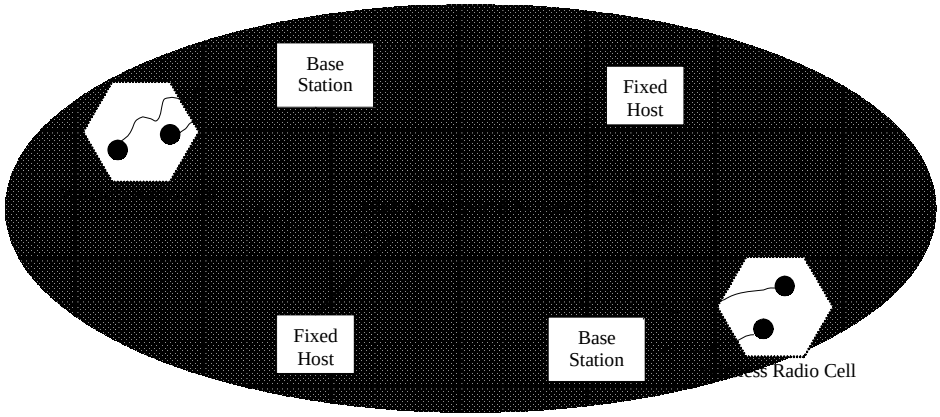
Given a number of identical broadcast channels and knowledge of the data item probabilities, how can we decide the contents of the multiple channels in order to reduce the average access time?

The aim of the present chapter is to provide an answer to the above question and it is organized as follows. "Background" describes in detail the assumed architectural model, whereas "Problem Formulation" defines the problem in mathematical terms. Optimal and heuristic algorithms for the problem of the data placement over multiple wireless channels are provided in "An Optimal Solution for Data Placement" and "Heuristic Approaches for Data Placement," respectively. Relevant work on multiple broadcast wireless channels is discussed in the next section, and the chapter concludes with a discussion about future trends and with a summary of its contributions.

Background

We consider a generic architecture of a mobile computing environment depicted in Figure 1 (Agrawal & Zeng, 2003; Dunham & Helal, 1995). Our system serves a geographical area, called the *coverage area*, where mobile clients or users (MUs) can move. The coverage

Figure 1. Generic architecture of a wireless system



area served by the wireless system is partitioned into a number of nonoverlapping regions, called *cells*. At the heart of the system lies a fixed backbone (wireline) network. A number of *fixed hosts* are connected to this network. Fixed hosts are general-purpose computers storing databases which are of interest to the mobile clients. Fixed hosts are not equipped to manage mobile units, although they can be configured to do so. Each cell is served by one *base station* (called simply *server*, in the sequel), which is connected to the fixed network and is equipped with wireless transmission and receiving capability. The server is responsible for converting the network signaling traffic and data traffic to the radio interface for communication with the mobile unit, and also for transmitting paging messages to the clients. Each server is assigned a number of identical broadcast channels and is able to transmit to all these channels concurrently.

In addition, we assume that every fixed host maintains a full replica of the database, and part of this database is broadcast by the server. The data are equal-sized items and the broadcast of each one of them takes one time unit, that is, a *tick*. The server has a priori knowledge of the client access probabilities and, based on these, it decides the contents of the broadcast, which is usually comprised of the most popular data items. Although, such a priori knowledge seems like a strong assumption, there are several methods for determining the data access probabilities. The works Sakata and Yu (2003) and Yu, Sakata, and Tan (2000) describe statistical methods for the estimation of these probabilities, whereas the works Nicopolitidis, Papadimitriou, and Pomportsis (2002) and Stathatos, Roussopoulos, and Baras (1997) present more practical approaches. The broadcast is a sequence of data blocks (containing data) and there are no index data.

Mobile clients are battery-powered, portable computers which use radio channels to communicate with the server to gain access to the fixed or wireless network. The clients have complete a priori knowledge of the broadcast schedule and they are able to listen simultaneously to all channels and perform instantaneous hopping from channel to channel. They continuously monitor the broadcast channels in order to acquire the data

of interest. We also assume that the size of a cell is such that the average time between when a client starts seeking for an item until it gets it from the broadcast is much smaller than the time required by the client to traverse the cell. Therefore, a user will seldom submit a query and exit a cell before receiving the response. The clients request one item per query.

Problem Formulation

Due to the nice characteristics of the generic paradigm of broadcast disks described in Acharya et al. (1995), that is, *fixed interarrival times* and *fixed-length broadcast cycle*, we adopt that model to implement in our broadcast channels. According to this model, the server partitions the database into groups and each group is broadcast into one channel. The contents of each group are cyclically broadcast in a round-robin fashion. Thus, we are left with the question of what to put into each group of items. In the next paragraphs, we will formulate this question into the problem of *Broadcast Program Generation on Multiple Channels*.

We consider a database D of equal-sized data items for which the access probabilities are given, and a set C of k broadcast channels ($C_1, C_2, ..., C_k$). Without loss of generality, we assume that the channels with small indexes will finally accommodate less items. We need to partition the items into these k wireless channels. Let our database be comprised of n items, that is, $D = d_1, d_2, \ldots, d_n$. The items are ordered from the most popular to the least popular and have access probabilities $P = p_1, p_2, \ldots, p_n$, respectively, where

$$\sum_{i=1}^{n} p_i = 1. \quad (1)$$

This ordering means that for any two coordinates (p_i, p_j) of the vector of probabilities P, where $1 \le i < j \le n$, it holds that $p_i \le p_j$. Since the items have equal size, the broadcast bandwidth needed to allocate each data item is the same and is denoted as a time slot. The interval between two consecutive broadcasts of the same item will be denoted as s_i, and its reciprocal as b_i. The latter estimates the probability that the item d_i will be selected for broadcast at each time slot. The average access time for item d_i will be denoted as a_i, whereas the average access time for all items at D will be:

$$a_{\text{total}} = \sum_{i=1}^{n} p_i * a_i. \quad (2)$$

Wong (1988) proved that, for the case of a single channel, when the data items are equisized, the average access time can be minimized if each data item is equally spaced and for any two items d_i and d_j, the following equation holds:

$$\frac{\sqrt{p_i}}{\sqrt{p_j}} = \frac{b_i}{b_j}. \tag{3}$$

We can easily show that in the case of k broadcast channels, the following holds: $\sum_{i=1}^{n} b_i = k$. Therefore, the analytical, minimum average access time for the database D broadcast on k channels is given by the following formula:

$$a_{\text{minimum}} = \frac{1}{2k}\left(\sum_{i=1}^{n}\sqrt{p_i}\right)^2, \tag{4}$$

assuming that the access frequencies p_i sum up to 1. Otherwise, the analytical, minimum average access time is given by the following formula:

$$a_{\text{minimum}} = \frac{1}{2k}\left(\sum_{i=1}^{n}\sqrt{\frac{p_i}{\sum_{j=1}^{n} p_j}}\right)^2. \tag{5}$$

This lower bound is difficult to achieve due to the difficulty in approximating Equation 3. Therefore, the average access time for each channel and, subsequently, the total average access time are different than that presented in Equation 5. Indeed, under the assumption that all interarrival times for an item d_i are equal, it follows that the average access time for d_i is

$$a_i = \frac{s_i}{2}, \tag{6}$$

and the average access delay for all items of channel C_{ij}, assuming that these are d_i, d_{i+1}, ..., d_j, will be:

$$a_{ij} = \frac{1}{2}(j - i + 1)\sum_{m=i}^{j} p_m \,,\ \mathrm{j} \geq \mathrm{i}\,. \tag{7}$$

Therefore, the total average access time will be

$$a_{\text{true\_total}} = \frac{1}{2}\sum_{m=1}^{k} N_m * P_m \,, \tag{8}$$

where N_m and P_m are the number of items and the popularity of the mth channel, respectively.

As mentioned above, generating a broadcast program for k channels can be viewed as a partition problem and can be characterized by an assignment $G:[1...n] \rightarrow [1...k]$ of items to the channels. Since more popular pages should be transmitted more frequently and, thus, should be accommodated into channels with a smaller number of items (i.e., channels with a small index), it is obvious that there can exist no pair d_i and d_j, with $i < j$, in the above ordering such that $G(d_i) > G(d_j)$. This means that either the two items will be accommodated into the same channel or the more popular of the two items will be accommodated into a smaller channel. Thus, creating a broadcast program for k channels is equivalent to determining a partition of the interval $[1...n]$.

Definition 1 (Broadcast Program Generation on Multiple Channels problem). Suppose that we have a database of n equisized items and a set of k broadcast wireless channels. Assume also that we are aware of the access frequency of each item. Then, the problem of *Broadcast Program Generation on Multiple Channels* is to find a partition of the n items into k groups and subsequently assign each group to a channel, such that the average access time for all items, that is, Equation 8, is minimized.

Optimal Solution for Data Placement

There exists an optimal solution to the problem described by Definition 1. This solution has been formulated into two different-in-nature algorithms. The first was presented in Peng and Chen (2003) and is based on the *A* optimization method*, whereas the second was presented in Yee, Omiecinski, and Navathe (2001) and is based on a dynamic programming approach. Denoting with $a_{\text{optimal\_minimal}}(i, j)$, the optimal solution (i.e., the minimal average access delay) for allocating items i to n on j channels, it is obvious that,

trivially, $a(i, 1) = a_{ij}$ (Equation 7). Then, the recurrence formula for the determination of the minimal average access delay is given by the following equation.

$$a_{\text{optimal\_minimal}}(i,k) = \min_{l \in \{i, i+1, \ldots, n\}} \{a_{il} + a_{\text{optimal\_minimal}}(l, k-1)\}. \quad (9)$$

We can prove that the following propositions hold.

Proposition 1. The time complexity of the dynamic programming algorithm is $O(k*n^2)$.

Proposition 2. The space complexity (storage of intermediate, partial solutions) of the dynamic programming algorithm is $O(k*n)$.

Obviously, the time complexity of the optimal dynamic programming solution is too high. In applications where the access frequencies change quite frequently or new items are to be broadcast, this approach turns out to be inapplicable. Therefore, various heuristics have been proposed in order to generate the partitioning very fast, producing average access time very close to the optimum. In the next section, we survey these approaches.

Heuristic Approaches for Data Placement

The procedure of determining a partition can be "top down," "bottom up," or "one scan." In the top-down (Hsu, Lee, & Chen, 2001; Peng & Chen, 2003; Yee, Navathe, Omiecinski, & Jermaine, 2002) approach, we start from a large partition, possibly including all the items, and gradually split it into smaller pieces. In the bottom-up (Hwang, Cho, & Hwang, 2001; Katsaros & Manolopoulos, 2004), we start with many small partitions which gradually grow, whereas the one-scan (Vaidya & Sohail, 1999) approach makes a single scan over the vector $P = p_1, p_2, \ldots, p_n$ of the access probabilities, assigning items to channels based on some criterion. The top-down and bottom-up make multiple passes over P (or parts of it) and make splitting or concatenating decisions based on the computation of Equation 8 over P (or portions of it).

The Bucketing Scheme

The *bucketing scheme* (Vaidya & Sohail, 1999) is the simplest approach for the assignment of items to channels. It makes a single scan over the vector P of access probabilities of the data and assigns them to channels based on the following criterion. Let $A_{\min}$ ($A_{\max}$) denote the minimum (maximum) value of $\sqrt{p_i}$, $1 \le i \le n$. Let $\delta = A_{\max} - A_{\min}$. If for the item

d_i it holds that $\sqrt{p_i} = A_{min}$, then d_i is assigned to channel C_k. Any other item d_i is assigned to channel C_{k-j} $(1 \leq j \leq k)$ if $(j - 1)*\delta/k < (\sqrt{p_i} - A_{min}) \leq (j*\delta/k)$. This partitioning criterion is suitable for the case when the access probabilities are uniformly distributed over the "probability interval" and performs poorly for skewed access patterns. On the other hand, this approach has the lowest complexity and never tries any "candidate" partitions in order to select the most appropriate.

Algorithm Bucketing (int *n*, int *k*, float vector *P*)

//*n*: number of items, *k*: number of channels, *P*: access probabilities

BEGIN

A_{min} = minimum($\sqrt{p_i}$), $1 \leq i \leq n$;

A_{max} = maximum($\sqrt{p_i}$), $1 \leq i \leq n$;

$\delta = A_{max} - A_{min}$;

for($i = 1$; $i \leq n$; $i = i + 1$){

    if($\sqrt{p_i} == A_{min}$) then

    item d_i is assigned to channel C_k;

else

    if($(j - 1)*\delta/k < (\sqrt{p_i} - A_{min}) \leq (j*\delta/k)$)

        item d_i is assigned to channel C_{k-j}, where $1 \leq j \leq k$;

}

END

Therefore, we can easily deduce the following proposition.

Proposition 3. The complexity of the bucketing scheme is $O(n)$.

The Growing Segments Scheme

The *growing segments* (Hwang et al., 2001) scheme starts with an initial "minimal" allocation assigning one item to each channel, which acts as the initial "seed" partition. Then, it enlarges each segment by including a number of items equal to the user-defined parameter *increment* and computes which of these enlargements gives the greatest reduction in average delay. Next, it selects the corresponding partition as the new seed

partition and continues until the partition covers the whole *P*. The parameter *increment* is very important and has the following trade-off associated with it: the greater the value of *increment*, the lower complexity the algorithm has and the lower quality the produced broadcast program has. Assuming that a set of points $S = (r_0, r_1, ..., r_k)$, satisfying the inequalities $0 = r_0 < r_1 < ... < r_k = n$, denotes a partition of the interval $(0, n]$, we have the following pseudocode for the growing segments algorithm.

Algorithm Growing Segments (int *n*, int *k*, float vector *P*)

//*n*: number of items, *k*: number of channels, *P*: access probabilities

BEGIN

// *getNextIncrement()*: returns an appropriate value for the next increment

// minDelay(); returns the minimum expected delay for a partition, i.e., Equation 8

increment = getNextIncrement();

$r_0 = 1$;

for($i = 1$; $i <= k$; ++ i) // initial setup of $(r_0, r_1, \ldots, r_k)$

    $r_i = r_{i-1}$ + increment;

while($r_k < n$){

    increment = getNextIncrement();

    for($i = 1$; $i <= k$; ++ i){

        $p_0 = r_0$;

        for($x = 1$; $x <= k$; ++ x)

            if ($x \geq i$) $p_x = r_x$ + increment;

            else $p_x = r_x$;

        $S_i = (p_0, p_1, \ldots, p_k)$;

    }

Find a partition $(r_0, r_1, \ldots, r_k)$ among S_is such that

    minDelay$(r_0, r_1, \ldots, r_k)$ = *min*{ *minDelay*(S_x) } for $x = 1, 2, ..., k$.

}

return $(r_0, r_1, \ldots, r_k)$;

END

Since the computation of Equation 8 takes $\Theta(n)$ time, and the algorithm makes $\Theta((n - k)/\text{increment})$ steps, computing at each step $\Theta(k)$ candidate partitions, we can easily deduce the following proposition.

Proposition 4. The complexity of the growing segments method is $O(n^2*(k/\text{increment}))$.

The Variant-Fan-Out Tree Scheme

The *variant fan-out with the constraint K* (VF^K) (Peng & Chen, 2003) scheme adopts a top-down approach. It starts with an initial allocation where all the items have been assigned to the first channel. Then repetitively, it determines which channel incurs the largest cost so far and partitions its contents into two groups. The partitioning is done by the routine *Partition*, which tries all possible partitions that respect the property that no channel can have more items than its next channel. The first group remains in the current channel and the newly created group is allocated to the next channel, shifting all the other channels downwards. This procedure repeats until all available channels are allocated. The cost of this algorithm is $O(k)$ times the cost of the *Partition* procedure. The cost of this procedure depends on the number of items of the channel that is to be partitioned, which in turn depends on the distribution of the access probabilities. Let the reduction in access delay, which will occur if we split a channel C_{ij} containing the items $d_i, d_{i+1}, \ldots, d_j$ into two channels C_{ip} and C_{pj} containing the items $d_i, d_{i+1}, \ldots, d_p$ and items $d_{p+1}, d_{p+2}, \ldots, d_j$, be denoted by $\acute{}(p)$, then $\acute{}(p) = a_{ij} - (a_{ip} + a_{pj})$. Then, the pseudocode for VF^K is shown on the next page.

Algorithm Variant Fan-out with the Constraint *K* (int *n*, int *k*, float vector *P*)

//*n*: number of items, *k*: number of channels, *P*: access probabilities

BEGIN

```
Create table AT with k rows;
AT(1).B = 1;        // AT(1).B records the beginning of channel
AT(1).E = n;        // AT(1).E records the end of channel
AT(1).LC = a_1n;    // AT(1).LC records the average access delay of channel
for(each row i in table AT AND i ≥ 2){
    AT(i).B = 0;  AT(i).E = 0;  AT(i).LC = 0;
}
pivot = 1;
repeat{
    Choose row i from table AT such that AT(i).LC is maximal among all unmarked rows;
    if (i == 1 or i == pivot){
        j = Partition(p_AT(i).B, p_AT(i).B+1, ..., p_AT(i).E);
        {Update table AT accordingly and unmark all rows; pivot ++;}
    }
    else{
        j = Partition(p_AT(i).B, p_AT(i).B+1, …, p_AT(i).E);
```

```
        if (AT(i – 1).E – AT(i – 1).B) < (j – AT (i).B)
            {Update table AT accordingly and unmark all rows; pivot ++;}
        else{
            Mark row i;
            Merge (p_AT(i).B, p_AT(i).B+1, ..., p_j ) with (p_j+1, p_j+2, ..., p_AT(i).E);
        }
    }
}until (pivot == k)
END
```

Procedure Partition $(p_i, p_{i+1}, ..., p_j)$.

BEGIN

Determine p^* such that $\delta(p^*) = \max_{\forall p \in \{i, i+(j-i+1)/2-1\}} \{\delta(p)\}$;

Assign items $p_{p^*+1}, p_{p^*+2}, ..., p_j$ into a new channel;

Return p^*;

END

Therefore, we can easily deduce the following proposition.

Proposition 5. The complexity of the VF^K method is $k^*(O(k^*\log(k)) + O(n))$.

The Greedy Scheme

The *greedy scheme* (Yee et al., 2002) adopts the top-down approach and it is very similar to the VF^K scheme. It performs several iterations. At each iteration, it chooses to partition the contents of the channel whose split will bring the largest reduction in access time. The partitioning point is determined by calling the routine *Partition* (see above). Thus, at each iteration, the greedy scheme computes (if not already computed) and stores the optimal split points for all channels that have not been split so far. Hence, it differs from VF^K in two aspects. First, it differs in the partitioning criterion (recall that VF^K splits the channel which incurs the largest access time). Second, after each split, it will compute and store the optimal split points of every channel.

Algorithm Greedy (int n, int k, float vector *P*)

//*n*: number of items, *k*: number of channels, *P*: access probabilities

```
BEGIN
numPartitions = 1;
while (numPartitions < k){
    for each partition r with data items i through j {
    // Find the best point to split in partition r
        for(s = i;  s < = j; s = s + 1)     // Initialize the best split point for this partition
                                            //as the ûrst data item. If we ûnd a better one
                                             //subsequently, update the best split point.
            if ( (s == i) OR  (localChange > C_ij^s) )
                localS = s;
                localChange = C_ij^s;   // Initialize the best solution as the one for the ûrst
                                        // partition. If we ûnd a better one subsequently,
                                        // update the best solution.
    if ( (r == 1) OR (globalChange > localChange) )
        globalChange = localChange;
        globalS = localS;
        bestpart = r;
    }
    split partition bestpart at point globalP;
    numPartitions = numPartitions + 1;
}
END
```

We can easily verify that the following proposition holds.

Proposition 6. The complexity of the greedy method is $O((n + k)*\log(k))$.

The Data-Based Scheme

The *data-based scheme* (DB; Hsu et al., 2001) is similar to VF^K, but avoids taking the local optimal decision of the *Partition* routine of VF^K, which splits a channel into two. DB has several phases. At each phase, it decides the contents of a particular channel starting from the smallest channel, which will accommodate the more frequently accessed data. First, it determines which is the maximum allowable number of items that can be

accommodated into the considered channel. This number can be computed with the help of Lemmas 1 and 2 (see also Hsu, Lee, & Chen, 2001).

Lemma 1. If we denote the number of items allocated to channel C_i as $|C_i|$, then it holds that $|C_1| \leq |C_2| \leq ... \leq |C_k|$.

Lemma 2. For the optimal allocation, it holds that $|C_{i-1}| \leq |C_i| \leq \frac{n - \sum_{j=1}^{i-1} |C_j|}{k - i + 1}$.

Then, it computes the average access delay for all the allowable allocations of items into the considered channel and selects the allocation with the minimum cost. The computation of the average access delay takes also into account the delay that will be incurred due to the items that will be allocated to the rest of the channels. This is the difference from VF^K. The above procedure continues until the allocation of all the items into the channels is completed.

Algorithm Data Based (int *n*, int *k*, float vector *P*)

//*n*: number of items, *k*: number of channels, *P*: access probabilities

BEGIN

for($i = 1$; $i \leq k - 2$; $i = i + 1$){

Calculate the range of the number of data items in channel *i*;

Determine the number of data items for allocating in channel *i*

such that the expected average access time is minimal;

}

Allocate the remainder data items into the last two channels so that

the average access time of all items is minimal;

END

Assuming that z_i is equal to the number of items allocated to channels C_1 to C_{i-1}, and y_i is equal to the range of items (as determined by Lemma 2), we can easily deduce the following proposition.

Proposition 7. The complexity of the data-based method is $\sum_{i=1}^{k} (y_i * (n - z_i))$.

The CascadedWebcasting Scheme

The *CascadedWebcasting scheme* (Casc; Katsaros & Manolopoulos, 2004) starts from a very basic intuition about the partitioning, claiming that there are three "classes" of items:

- a practically constant number of items with *high probability*,
- items belonging to a few *large groups*, and
- leftover items, which contribute *negligibly* to the total delay.

Using this intuition about the partitioning, Casc is a bottom-up scheme that uses "predetermined" initial seed partitions, which subsequently are greedily concatenated until the number of partitions becomes equal to the number of available broadcast channels. Initially, the seed partitions $P_1^0, P_2^0, ..., P_\nu^0$ are generated, with sizes equal to 2^0, 2^1, 2^2, ... Then, $\nu - k$ merging steps ($\lceil \nu = \log_2(n+1) \rceil$) are performed. At the ith merging step ($1 \le i \le \nu - k$), the algorithm tries $\nu - i - 1$ concatenations and selects the one which incurs the least expected cost. Due to the way the partitions are concatenated and the size of the initial seed partitions, it is obvious that at each step of the algorithm, the size of a partition is always smaller than the size of its successive partition, thus respecting Lemma 1. ™f the number n of items is not equal to $2^\nu - 1$, but it is $2^\nu - 1 < n\,(= 2^\nu - 1 + \beta^{2}) < 2^{\nu+1} - 1$ for some $\beta > 0$, then we treat the first $2^\nu - 1$ items with the procedure mentioned above and simply append the last ^{2} items to the last channel. The pseudocode for Casc is presented below.

Algorithm CascadedWebcasting (int n, int k, float vector P)

//n: number of items, k: number of channels, P: access probabilities

BEGIN

$v = \lceil \log_2(n+1) \rceil$;

create v seed partitions, $P_1^0, P_2^0, ..., P_v^0$ with $size(P_i^0) = 2^{i-1}$;

$P_l = P - (P_1^0 \cup P_2^0 \cup , ..., \cup P_v^0)$;

$P^0 = \{P_1^0, P_2^0, ..., P_v^0\}$; //the set of seed partitions

for($i = 1$; $i \le v - k$; $i = i + 1$){ //perform $v - k$ merging steps

for($j = 1$; $j \le v - i - 1$); $j = j + 1$)

$C^j = \{C_1^i(= P_1^{i-1}), \ldots, C_j^i(= merge(P_j^{i-1}, P_{j+1}^{i-1})), ..., C_{v-i-1}^i(= P_{v-i}^{i-1})\}$;

$P^i = minDelay(C^j)$; //The partition incurring the minimum delay

}

$P^{final} = \{P_1^{½-k}, ..., P_k^{½-k} \cup P_l\}$;

return P^{final};

END

It is easy to prove the following proposition (see Katsaros & Manolopoulos, 2004)).

Proposition 8. The complexity of Casc is dominated by $O(n)$.

A comprehensive performance evaluation of the aforementioned algorithms has been conducted in Katsaros and Manolopoulos (2004). There, it was recognized that the greedy scheme, VF^K, and DB perform very close to optimum with respect to the reduction of the average access delay, but they incur significant execution cost. The fastest running algorithms are bucketing and Casc, with the latter producing partitions not far from the optimum with respect to the average access delay.

Relevant Work on Data Broadcasting over Multiple Wireless Channels

There are quite a lot of research efforts in finding efficient methods for broadcasting dependent data on multiple (or single) wireless channels. We mention the most important of them. In Chehadeh, Hurson, and Kavehrad (1999), Chung and Kim (2001), Lee and Lo (2003), Lee, Lo, and Chen (2002), and Liberatore (2004), the issue of scheduling dependent data on a single broadcast channel is investigated, whereas in Huang, Chen, and Peng (2003) and Juran, Hurson, Vijaykrishnan, and Kim (2004), the issue of dependent data scheduling over multiple channels is investigated.

Power conservation is a key issue for mobile computers. Air-indexing techniques for the broadcast data can be employed in order to help clients reduce the time they remain tuned into the broadcast channel(s). Various indexing techniques have been proposed and some of them have been described in chapters of this book.

The *index tree* (Imielinski et al., 1997) is the application of the traditional disk-based B-tree indexing technique in the context of wireless channels. Also, the well-known *signature tree* indexing method has been proposed for the case of broadcast data indexing (Lee & Lee, 1996). An amalgamation of the aforementioned techniques, the *hybrid index tree*, is proposed in Hu, Lee, and Lee (2001). The aforementioned techniques proposed balanced indexing structures. To achieve better performance for skewed queries, various techniques investigated the approach of creating unbalanced indexes. For instance, the index proposed in Shivakumar & Venkatasubramanian (1996) is a generalization of the *binary alphabetic tree* requiring that the number of available wireless channels is equal to the number of tree levels, whereas a generalization of the classic *binary Huffman tree* for the case of broadcast channels was proposed in Chen, Wu, and Yu (2003).

While many studies, like ours, consider data scheduling and indexing separately, some works addressed the issue of allocation of both data and index over multiple channels. In Prabhakara et al. (2000), various broadcast and client access methods are proposed, whereas the works appearing in Lo and Chen (2000) and Hsu, Lee, and Chen (2002)

examined the issue of allocating index and data at the same time, so that the average access time is minimized. Finally, an interesting indexing structure for multiple broadcast channels is presented in Lee and Jung (2003).

Conclusion

Although the concept of broadacast delivery is not new, recently, the dissemination of data items by broadcast channels has attracted considerable attention due to the excellent scalability it offers. There are many scenarios where a server has access to multiple low-bandwidth physical channels which cannot be combined to form a single high-bandwidth channel. Possible reasons for the existence of multiple broadcast channels include application scalability, fault tolerance, reconfiguration of adjoining cells, and heterogeneous client-communication capabilities. Thus, it becomes a necessity to devise appropriate methods for data allocation over multiple wireless channels.

The present chapter explored the issue of generating broadcast programs for multiple wireless channels when the data access probabilities and the number of wireless channels are given, assuming that there are no dependencies among the data to be broadcasted. Initially, we gave the mathematical formulation for the problem and presented an optimal solution for it based on dynamic programming. Then, we presented six heuristic approaches which present different trade-offs between optimality in terms of average access delay and execution time.

Acknowledgments

This work has been funded through the bilateral program of scientific cooperation between Greece and Turkey (Γ.Γ.Ε.Τ.) and from TUBITAK Grant No. 102E021).

References

Acharya, S., Alonso, R., Franklin, M. J., & Zdonik, S. B. (1995). Broadcast disks: Data management for asymmetric communications environments. *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 199-210.

Agrawal, D. P., & Zeng, Q.-A. (2003). *Introduction to wireless and mobile systems, Florance:* Brooks/Cole (Thomson Learning Inc.).

Aksoy, D., & Franklin, M. J. (1999). RxW: A scheduling approach for large-scale on-demand data broadcast. *IEEE/ACM Transactions on Networking, 7*(6), 846-860.

Armon, A., & Levy, H. (2004). Cache satellite distribution systems: Modeling, analysis, and efficient operation. *IEEE Journal on Selected Areas in Communications, 22*(2), 218-228.

Chehadeh, Y. C., Hurson, A. R., & Kavehrad, M. (1999). Object organization on a single broadcast channel in the mobile computing environment. *Multimedia Tools and Applications, 9*(1), 69-94.

Chen, M.-S., Wu, K.-L., & Yu, P. S. (2003). Optimizing index allocation for sequential data broadcasting in wireless mobile computing. *IEEE Transactions on Knowledge and Data Engineering, 15*(1), 161-173.

Chung, Y. D., & Kim, M.-H.(2001). Effective data placement for wireless broadcast. *Distributed and Parallel Databases, 9*(2), 133-150.

Datta, A., Vandermeer, D. E., Celik, A., & Kumar, V. (1999). Broadcast protocols to support efficient retrieval from databases by mobile users. *ACM Transactions on Database Systems, 24*(1), 1-79.

Dunham, M. H., & Helal, A. (1995). Mobile computing and databases: Anything new? *ACM SIGMOD Record, 24*(4), 5-9.

Hsu, C.-H., Lee, G., & Chen, A. L. P. (2001). A near optimal algorithm for generating broadcast programs on multiple channels. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 303-309.Hsu, C.-H., Lee, G., & Chen, A. L. P. (2002). Index and data allocation on multiple broadcast channels considering data access frequencies. *Proceedings of the International Conference on Mobile Data Management (MDM)*, 87-93.

Hu, Q. L., Lee, W.-C., & Lee, D. L. (2001). A hybrid index technique for power efficient data broadcast. *Distributed and Parallel Databases, 9*(2), 151-177.

Huang, J.-L., Chen, M.-S., & Peng, W.-C. (2003). Broadcasting dependent data for ordered queries without replication in a multi-channel mobile environment. *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, 692-694.

Hwang, J.-H., Cho, S., & Hwang, C.-S. (2001). Optimized scheduling on broadcast disks. *Proceedings of the international conference on mobile data management (MDM)* (pp. 91-104).

Iacono, A. L., & Rose, C. (2000). Infostations: New perspectives on wireless data networks. In S. Tekinay (Ed.), *Next Generation Wireless Networks* (Vol. 598). NJ: Kluwer Academic Publishers.

Imielinski, T., Viswanathan, S., & Badrinath, B. R. (1997). Data on air: Organization and access. *IEEE Transactions on Knowledge and Data Engineering, 9*(3), 353-372.

Juran, J., Hurson, A. R., Vijaykrishnan, N., & Kim, S. (2004). Data organization and retrieval on parallel air channels: Performance and energy issues. *ACM/Kluwer Wireless Networks, 10*(2), 183-195.

Katsaros, D., & Manolopoulos, Y. (2004). Broadcast program generation for webcasting. *Data and Knowledge Engineering, 49*(1), 1-21.

Lee, B., & Jung, S. (2003). An efficient tree-structured index allocation method over multiple broadcast channels in mobile environments.. *Proceedings of the database and experts systems applications workshop (DEXA)* (pp. 433-443).

Lee, G., & Lo, S.-C. (2003). Broadcast data allocation for efficient access of multiple data items in mobile environments. *ACM/Kluwer Mobile Networks and Applications*, *8*(4), 365-375.

Lee, G., Lo, S.-C., & Chen, A. L. P. (2002). Data allocation on wireless broadcast channels for efficient query processing. *IEEE Transactions on Computers*, *51*(10), 1237-1252.Lee, W.-C., & Lee, D. L. (1996). Using signature techniques for information filtering in wireless and mobile environments. *Distributed and Parallel Databases*, *4*(3), 205-227.

Liberatore, V. (2004). Circular arrangements and cyclic broadcast scheduling. *Journal of Algorithms*, *51*(2), 185-215.

Lo, S.-C., & Chen, A. L. P. (2000). Optimal index and data allocation in multiple broadcast channels. *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, 293-302.

Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2002). Using learning automata for adaptive push-based data broadcasting in asymmetric wireless environments. *IEEE Transactions on Vehicular Technology*, *51*(6), 1652-1660.

Peng, W.-C., & Chen, M.-S. (2003). Efficient channel allocation tree generation for data broadcasting in a mobile computing environment. *ACM/Kluwer Wireless Networks*, *9*(2), 117-129.

Prabhakara, K., Hua, K. A., & Oh, J. (2000). Multi-level multi-channel air cache designs for broadcasting in a mobile environment. *Proceedings of the IEEE Conference on Data Engineering (ICDE)*, 167-176.

Sakata, T., & Yu, J. X. (2003). Statistical estimation of access frequencies: Problems, solutions and consistencies. *ACM/Kluwer Wireless Networks*, *9*(6), 647-657.

Shivakumar, N., & Venkatasubramanian, S. (1996). Efficient indexing for broadcast based wireless systems. *ACM/Baltzer Mobile Networks and Applications*, *1*(4), 433-446.

Stathatos, K., Roussopoulos, N., & Baras, J. S. (1997). Adaptive data broadcast in hybrid networks. *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*, 326-335.

Vaidya, N., & Sohail, H. (1999). Scheduling data broadcast in asymmetric communication environments. *ACM/Baltzer Wireless Networks*, *5*(3), 171-182.

Wong, J. W. (1988). Broadcast delivery. *Proceedings of the IEEE*, *76*(12), 1566-1577.

Yee, W. G., & Navathe, S. B. (2003). Efficient data access to multi-channel broadcast programs. *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 153-160.

Yee, W. G., Navathe, S. B., Omiecinski, E., & Jermaine, C. (2002). Bridging the gap between response time and energy-efficiency in broadcast schedule design. *Proceedings of the international conference on extending data base technology (EDBT)* (pp. 572-589).

Yee, W. G., Omiecinski, E., & Navathe, S. B. (2001). *Efficient data allocation for broadcast disk arrays* (Tech. Rep. No. GIT-CC-02-20). Georgia Institute of Technology, Atlanta.

Yu, J. X., Sakata, T., & Tan, K.-L. (2000). Statistical estimation of access frequencies in data broadcasting environments. *ACM/Baltzer Wireless Networks, 6*(2), 89-98.

Section II.

Location Management